Projekt 2
EIT060

Alexander Najafi (elt12ana)
Tobias Mähl (har11tma)
Simon Rydebrink (dat14sry)
Sebastian Karabeleski (dat12sk1)

2016-03-03

Innehåll

1 Introduktion

Denna rapport presenterar en lösning för att hantera ett journaler på ett sjukhus av ett säkert system. Vi kommer även tydligt och genomgående beskriva vår implementering av detta system.

En simulerad databas kommer att användas för att förvara journalerna. De olika sorters roller vi tagit hänsyn till när vi utformade systemet är:

- Patient: En patient kan ha en eller fler journaler i systemet och kommer alltid att ha rättigheten att läsa dem.
- Läkare: En läkare får både läs- och skrivrättigheter till alla sina patienters journaler samt även läsrättigheter till alla patienters journaler inom sin avdelning. Endast en läkare har också möjligheten att skapa en ny journal till en patient.
- Sjuksköterska: En sjuksköterska kommer att ha läs- och skrivrättigheter till alla patienter registrerade till sig och även sjuksköterskan har läsrättigheter till patienters journaler inom sin avdelning.

2 Arkitekturell överblick

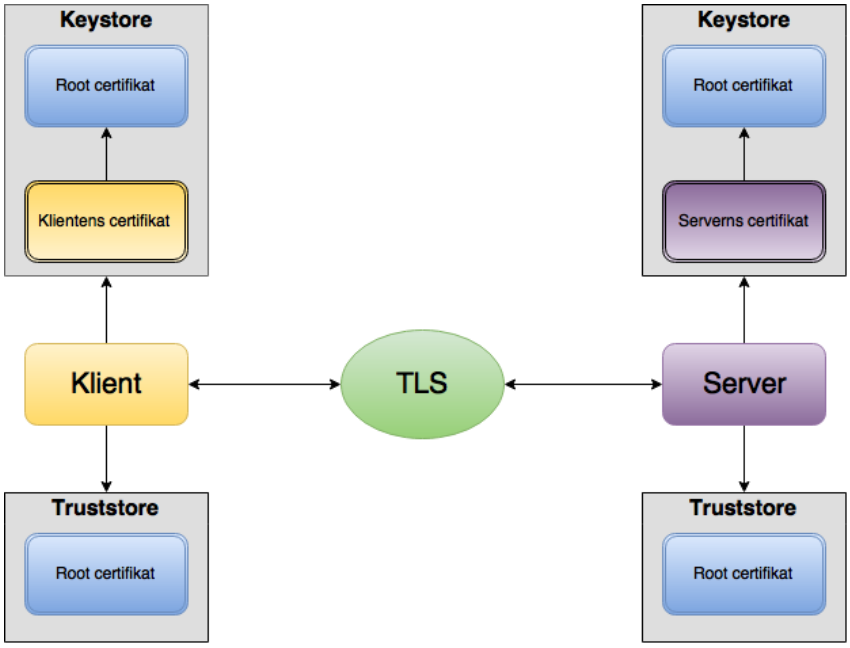
Implementationen av journalsystemet för denna uppgift implementerades med hjälp av en central server och en implementation av en klientmjukvara. Klientmjukvaran skulle kunna vara implementerad på många olika sätt och plattformar så som för t.ex. mobiltelefoner men för denna uppgift valdes det att implementera en klient för datorer som var helt terminalbaserad. För kommunikationen mellan servern och klienten utvecklades en specifikation för ett JSON protokoll. JSON, som är ett open-source program framtaget för att automatisera parsing av meddelanden, implementerades för att kunna sköta kommunikationen på ett enkelt och strukturerat sätt. Det valdes för att standarden redan används flitigt och det finns många implementationer av parsing för många olika plattformar, vilket i sin tur skulle kunna göra att klienter för fler plattformar utvecklades.

2.1 Server

2.1.1 Överblick

Servern är den centrala delen i journalsystemet. Det är servern som håller reda på alla patienter, doktorer, sköterskor, journaler, inloggningar osv. I servern finns en implementation av en databas som innehåller alla journaler, användare osv. I implementationen implementerades databasen helt i internminnet för denna uppgift. Varje gång servern startar upp stoppas det in testdata i dess databas. Ingen information som sparas i databasen finns kvar om den skulle startas om. I ett verkligt system skulle så klart detta kunna implementeras med hjälp av t.ex. en MySQL databas.

För att kunna kommunicera med servern måste en klient autentisera sig. Detta sker genom en tvåstegsverifikation där det ena steget är en autentisering med SSL-certifikat och den andra är med ett användarspecifikt lösenord. Säkerhetsaspekterna och detaljerna kring detta beskrivs nedan.



Figur 1: Diagram över hur systemet använder SSL.

För att en klient ska kunna skicka förfrågningar till servern om att utföra uppgifter måste denna autentisera sig. Efter en lyckad autentisering genereras en token av servern som skickas tillbaka till klienten. Denna token måste skickas med varje förfrågan som klienten gör till servern. På så sätt kan servern veta precis vem som gör en förfrågan och baserat på detta bestämmer servern om personen får lov att utföra uppgiften (som t.ex. kan vara att skapa en ny patient, uppdatera patientuppgifter eller skapa en artikel i en patients journal) som specificeras.

I servern implementerades en statisk referensmonitor för att sköta kontrollen om en användare får utföra en viss uppgift. När en förfrågan kommer in till servern tar denna först reda på vilken typ av uppgift den ska utföra. Efter detta kontrollerar den om användaren som försöker utföra uppgiften har rätt att göra detta. Om kontrollen går igenom så utförs uppgiften och servern skickar tillbaka ett passande svar till klienten.

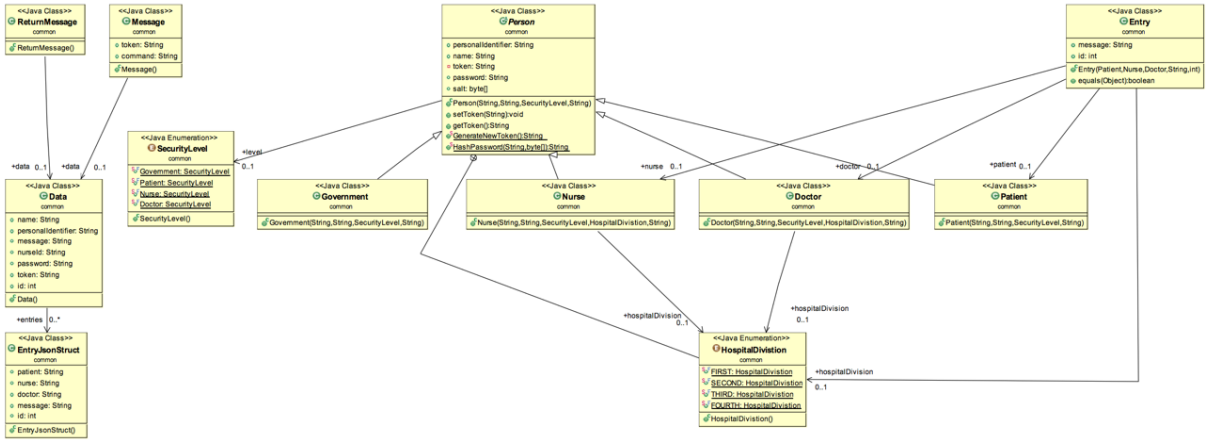
Vid uppkoppling mellan servern och klienten går det till som enligt nedan. Klienten skickar en förfrågan till servern som svarar med sitt certifikat. Klienten verifierar certifikatet eftersom det är signerat av en tillförlitlig CA och skickar tillbaka sitt eget certifikat. Servern verifierar klientens certifikat och uppkopplingen räknas nu som säker och delandet av information kan börja. I detta utbyte av information har klient och server kommit överens om vilken krypteringsalgoritm som skall användas för att dela information.

2.1.2 Implementation

I UML-diagrammet nedan syns de klasser som är essentiella för servern. Servern ser alla användare som personer, skillnaden mellan de olika personerna så som, doktorer, patienter och sköterskor är dess roll i systemet. De olika användarna har olika roller, och olika roller får göra olika uppgifter. Vilka uppgifter som vilka roller får utföra sköts av referensmonitorn. Till skillnad från patienter och statliga organisationer så har doktorer och sköterskor även en avdelning som tillhör deras användare.

Programmet består av tre paket. UML-diagrammet visar visar common-paketet. Ett paket med klasser som både servern och klienten känner till. Common-paketet hanterar kommunikationen och mellan server och klient och där ligger även den intelligens som används av både server och klient. I server-paketet och klient-paketet ligger intelligens som är specifik för respektive funktion. Tillexempel hur klienten skall

skicka användarens input till servern, och hur servern skall ta emot och behandla den info som klienten skickar.



Figur 2: UML diagram över servermjukvaran.

2.2 Textbaserad klient

Klienten som implementerats för denna uppgift är en enkel, helt textbaserad klient med syfte att demonstrera servens implementation. När klienten startas ansluter denna till servern och autentiserar sig själv med hjälp av ett x509 certifikat. När en användare sedan använder klienten för att logga in skriver denna in sitt personnummer och lösenord som skickas till servern. Vid en lyckad autentisering gäller det för klienten att hålla koll på den token som servern då skickar tillbaka.

Klienten tar inmatning från användaren i form av ett komando och ber sedan om ytterligare information som behövs för det valda komandot. När användaren matat in all information som behövs, så skapar klienten ett JSON meddelande som skickas till servern. När klienten får ett svar från servern tolkar den detta och ger ett användarvänligt meddelande till användaren. När användaren bestämmer sig för att logga ut skickar klienten ett meddelande till servern om att användaren ska loggas ut. Efter detta är den token som klienten använt under sessionen värdelös och kan inte användas mer. För att göra fler förfrågningar till servern måste en ny token genereras av servern genom att en användare loggar in igen.

3 Etisk diskussion

För att en vettig diskussion i etik och moral angående journalsystemet ska kunna genomföras måste det först etableras var denna fråga skapas.

Patienter tar sig till en läkare eftersom de har ett problem som de hoppas läkaren kan hjälpa till att korrigera. Läkaren ska då genomföra vården av patienter med avsikt och skyldighet att göra alla rimliga ansträngningar för att hjälpa dem. Men varje medicinskt fall är inbäddad i ett större sammanhang för de personer och institutioner som är inblandade. Detta större sammanhang påverkas av de beslut som fattas av eller om patienten. Dessa beslut har psykologiska, känslomässiga, ekonomiska, juridiska, vetenskapliga och religiösa påverkan på andra som kan vara av avgörande betydelse för patienten. Därför sparas medicinska journaler med omfattande säkerhet, för om någon utomstående skulle få kännedom om dessa beslut, kan det få betydande konsekvenser de inblandade.

Under designstadiet med kunden bör det informeras var säkerhetsstandarden enligt Personuppgiftslagen ligger och vilken miniminivå säkerheten på systemet bör hålla. Då kan det också rekommenderas och diskuteras med kunden vad som kan vara lämpligt utöver denna. Detta beroende på vilka risker som finns redan nu och vilka som kan bli aktuella i framtiden. Men i slutändan så är det alltid kunden som

måste ta ett beslut då det är dem som ska använda och betala för det. Ditt ansvar som ingenjör är att informera och rekommendera så att det bästa beslutet för kunden kan tas.

För att uppnå en design för att ge användare olika behörigheter i ett journalsystem, så hade vi valt att ge alla användare väldigt fördefinierade roller. För att tillämpa "principle of least privilege" på ett smidigt och enkelt sätt. Försöka undvika specialfall och hålla administrationen av systemet så enkel som möjlig.

De olika aktörerna vid upphandling och användning av ett sjukhusjournalsystem har alla lite olika etiska dilemma de står inför. När systemet ska köpas upp måste sjukhusledningen se till att det uppfyller alla lämpliga lagar för att hantera den aktuella information, det vill säga Personuppgiftslagen etc. Utöver det så måste de väga ekonomiska fördelar mot olika säkerhetsaspekter för systemet. För ingenjören/designern/implementeraren så kan det vara smidigt att inte lämna över all access till systemet när det anses färdigt. Utan att ha kvar en bakdörr eller liknande för att kunna rätta till saker i efterhand. Även om det skulle vara smidigt så är det tillgänglighet mot integritet. Medans för sjukhuspersonalen så gäller det att inte utnyttja dem designvalen som har gjorts för systemet. Detta även om dem skulle ha tillgång till någon typ av information enligt systemet så betyder inte det alltid att det är etiskt rätt att använda den. Sen om dem skulle finna buggar eller liknande som kringgår säkerhetsaspekterna under användning, så kan det tyckas att dessa ska rapporteras och inte utnyttjas.

3.1 Rättigheter

I ett fiktivit, optimalt system skulle en rättighetstabell kunna se ut som den nedan.

Person	**Läsa**	**Skriva**	**Ändra**	**Radera**
Patient	Ska kunna läsa all info om sig själv	Ej	Ej	Ej
Sköterska	Ska kunna läsa och skriva journaler på patienter som är inskrivna på sjukhuset eller de som de någon gång behandlat		Ska kunna ändra det som han skapat	Ej
Läkare	Ska kunna läsa och skriva journaler på patienter som är inskrivna på sjukhuset eller de som de någon gång behandlat		Ska kunna ändra det som han skapat	Ej
Myndighet	Ej	Ej	Ej	Ja, vad som helst

4 Säkerhetsutvärdering

För att hålla systemet säkert har vi tagit flertalet åtgärder. Den första är att uppkopplingen mellan klienten och servern kräver en SSL uppkoppling. Detta uppnås genom att server och klient båda litar på samma certifikat. Detta är vår ena faktor i vår tvåfaktorsautentisering. SSL uppkopplingen låter oss skicka information mellan klient och server på ett säkert sätt och vi kan vara säker på att informationen når fram till rätt server samt att informationen kommer fram oläst och oförändrad. Certifikaten är unika för varje användare, användaren får ha sitt certifikat lagrat på t.ex. ett USB-minne eller i sin dator för att lösningen ska kunna fungera.

Uppkopplingen använder SHA1 med RSA som signaturalgoritm. Detta är inte den absolut säkraste uppkopplingen, men den uppfyller enligt oss rimliga säkerhetskrav. SHA1 är i teorin knäckt då det skulle vara möjligt att hitta en kollision i hashningen men detta är ännu inte genomfört rent praktiskt.

När en säker uppkoppling är upprättad når vi steg två i vår tvåfaktorsautentisering. Användaren ombeds nu att ange sitt användarnamn och lösenord. När användaren är inloggad vet systemet vilka rättigheter användaren har och ger då användaren rättigheter kopplade till dennes behörighet till de funktioner som den bör ha.

Lösenordet får användaren välja själv. Men uppmanas att välja ett lösenord som innehåller både gemener, versaler, samt siffror eller specialtecken. Detta innebär en svaghet och en säkerhetsrisk i vårt program, då ett dåligt valt lösenord skulle möjliggöra en hackare att utföra en brute force attack, dictionary attack eller en rainbow table attack på en rimlig tid.

När en säker uppkoppling har etablerats och en användare har autentiserat sig själv tilldelas denne en token (en lång slumpmässigt vald sträng), en token som användaren kommer att skicka med i alla förfrågningar som den gör. När servern mottager en förfrågan, till exempel att skriva ett nytt journalinlägg så kontrollerar serven den token som användaren har skickat med.

4.0.1 Styrkor och svagheter

Systemet för journaler har utvecklats på sådant vis så att följande attacker *inte* kan utföras på systemet.

Man-in-the-middle attack Att försvara sig mot Man-in-the-middle attacker är väldigt svårt utan en separat, säker, kanal för verifiering. Genom att använda SSL-anslutningar och certifikat, som är signerade av gemensam certifikatutfärdare(CA), så går denna attacken också att undvika. En "eavesdropper" hade behövt lura CA:n att signera sitt certifikat med serverns namn. Detta borde inte vara möjligt då CA:n borde kolla alla certifikat den signerar.

Eavsdropping Denna attacken undviks helt enkelt genom att använda SSL-protokollet.

Fake Certifkat Eftersom vi använder en enda certifkatsutfärdare(CA) som signerar alla certifikat, så blir inga certifikat signerade av någon annan.

Brute-Force attack Såvitt man hållit sig till våra förslag om hur ditt lösenord ska vara utformat så borde inte brute-force attacker vara en fara. Med de specifikationer vi utformade tar det mer än ett år innan man kan knäcka lösenordet med en brute-force attack och då måste ändå certifikatet förnyas.

Spoofed Client Om en anfallare skriver en låtsasklient och lurar en användare, låt oss kalla henne Alice, att använda den så har vi inget skydd mot det. Det allvarligaste en anfallare kan göra i detta fallet är att stjäla all patientinformation som Alice har åtkomst till. Oturligt nog det enda är att försäkra sig att man har laddat ner rätt klient är att besöka sjukhuset hemsida genom en SSL-anslutning eller på något annat säkert vis.

Dictionary attack Om man håller sig undan att använda vanliga ord blir en dictionary attack meningslös. Det är däremot upp till användaren att välja sitt lösenord och vi kräver inte hur de utformar sitt lösenord. Man skulle kunna lägga till att man bara får t.ex. skriva in sitt lösenord var femte sekund, och då tar den attacken längre tid.

4.1 Journalernas tillgänglighet

Inledningsvis, så förser vi konfidentialitet till våra patienter. Konfidentialitet innebär skyddet av personlig data och detta förses bara genom att låta auktoriserade användare läsa de olika journalerna. Dessutom, när man skapar en journal, så är det enda som kan ändras av en auktoriserad användare är patientens information, inte av den som behandlar den, vilket tillför integritet. Detta förhindrar folk från att ändra vem som behandlar patienten till sin egen fördel. T.ex. för att en icke auktoriserad användare tillåts komma åt det.

5 Appendix

5.1 Peer-reviews

Nedan finns de peer reviews som inkommit från de andra grupperna.

Introduktion
Tydlig introduktion som ger en god överblick.

Arkitekturell överblick
Ingående beskrivning av hur systemet är uppbyggt.
Kan vara värt att nämna hur lösenord lagras på servern.
Hade varit trevligt med någon mening om vad ett JSON-meddelande är.

Etisk Diskussion
Utförligt resonemang kring leverantörens (ingenjören) skyldigheter vad gäller designbeslut samt att hanteringen av information sker enligt personuppgiftslagen.

Kunde varit ett mer utförligt resonemang kring nedanstående punkt,

- Formulate an access control scheme (define it, do not implement it) that provides the best tradeoff between confidentiality, integrity and availability.

Säkerhetsutvärdering
“då ett dåligt valt lösenord skulle möjliggöra” ….?

God insikt i vilka svagheter samt styrkor som finns i implementationen.

Generellt
Generellt sett behöver texten korrekturläsas en gång, det finns en del stavfel.

High-level architectural overview
In the initial high-level explanation of the system you provide a brief overview of your system with a good explanation and argument for your design choices. Here we also found a spelling error: "*Det valdes för att standarden redan används flitigt och det finns många implementationer av parsning för många olika plattformar, vilket i sin rut skulle kunna göra att klienter för fler plattformar utvecklades.*".

You then move on to explain your server a little more closely. This is also done mostly well but there are some small improvements to be made. The instruction states someone who hasn't taken the course should be able to understand this part. In the closing part of the server explanation you mention a static reference monitor, maybe a small explanation of what this is in layman terms would be a good idea. Spelling error: "*Detta sker genom en tvåstegs-verifikation där det ena steget är en autentisering med ssl-vertifikat och den andra är med ett användar-spevifikt lösenord.*". Fast fingers? Proofread.

You then go on to explain your implementation, here i have trouble finding a graphical representation of your keystores and truststores as required in the instruction 3.1. The rest of this explanation is easy to understand and lists the main parts of your implementation.

Finally you explain your client in a brief and easy way. You mention the way of authentication as well as how the user is supposed to interact with the system. Considering that the project isn't about the client in any bigger way than the connection this seems enough to us.

Ethical discussion
Here you bring up many good points about why the confidentiality of the system is of such importance. You seem very adamant about the importance of the conceiving part of the project alongside the customer, which we agree on. There does however seem to be no clear coverage of the task: *"How would you implement access control that is suitable for live production in a real hospital environment? Formulate an access control scheme (dene it, do not implement it) that provides the best tradeoff between confidentiality, integrity and availability. Compare this access control scheme to the one you have implemented here and list advantages and drawbacks of each scheme.*". However apart from this the rest of the discussion is both relevant and thorough.

Security evaluation
Here you mention many different kinds of attacks and exploits as well as how to avoid or completely negate them as required by the instructions. You also mention your cipher suites and argue why you use them as well as mention what level of security they can be expected to provide. We see no issues here.

5.2 Gjorda förbättringar

Efter att ha mottagit rapporter om förbättringar från två andra grupper så har vi främst förbättrat språkliga misstag så som stavfel och meningsbyggnader. Vi har förbättrat vår rapport på så sätt att vi har diskuterat och skapat en tabell över ett optimalt kontrolschema för ett system på ett verkligt sjukhus.

Efter att ha haft en code review med en annan grupp så gjorde vi inga ändringar förutom ett utskriftsmeddelande som var fel i klienten. Vi ordnade även så att lösenord som skrivs in i terminalen maskeras och inte syns.

Vi har uppdaterad UML diagramet i rapporten efter den senaste versionen av koden.

5.3 Code review och Project contributions

Nedan finns dokument för code review och project contribution.

LUNDS UNIVERSITET
Lunds Tekniska Högskola

EIT060 Computer Security
Code Review Form
Spring 2016

How to perform the review

The group whose code is being reviewed (reviewee) is denoted 'Group E' in the sequel. The group that is performing the review (reviewer) is denoted 'Group R'. The review process is as follows.

1. Group E demonstrates that their TLS connection is fully functional.
2. Group E demonstrates their two-factor authentication.
3. Group E demonstrates that their access control scheme is implemented correctly.
   Group R specifies at least one positive and one negative usage scenario per user type. That is, one usage scenario that should work (doctor adding a record to his patient) and one that should not (patient deleting a record).
4. Group E demonstrates their audit log functionality.

Review report

Group R fills out the relevant information in this section.

Group R number: 24 Group E number: 23

Group R hereby confirms that the project implementation of Group E is of sufficient quality and complies with the project requirements.

Group R signatures:

Lund, 2016-03-01

LUNDS UNIVERSITET
Lunds Tekniska Högskola

EIT060 Computer Security
Contribution Statement Form
Spring 2016

Individual project contributions

Group number: 23

name	contribution
Simon Rydebrink	All parts
Sebastian Kacabelesler	All parts
Tobias Mähl	All parts
Alexander Najafi	All parts

All group members have actively taken part in and contributed to the project in sufficient part.

Member signatures:

Lund, 2016- 03-03